From Programs to Games: Invariance and Safety for Bisimulation

Marc Pauly

Center for Mathematics and Computer Science (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
pauly@cwi.nl

Abstract. Bisimulation is generalized from process models to game models which are described using *Game Logic* (*GL*), a logic which extends Propositional Dynamic Logic by an additional operator *dual* which allows for the construction of complex 2-player games. It is shown that bisimilar states satisfy the same *GL*-formulas (invariance), and that an atomic bisimulation can be lifted to non-atomic *GL*-games (safety). Over process models, *GL* forms a highly expressive fragment of the modal μ-calculus, and within first-order logic, the game operations of *GL* are complete: they suffice to construct all first-order definable games which are monotonic and safe for bisimulation.

1 Introduction

Among the different notions of process equivalence one can consider, bisimulation has received much attention especially within the logic community. From the perspective of modal logic, there is a tight correspondence between bisimilar states of a process (Kripke model) and states which make the same modal formulas true: Bisimilar states satisfy the same modal formulas, and for certain classes of Kripke models (e.g. finite models), the converse holds as well. This bisimulation-invariance result makes bisimulation an attractive notion of equivalence between Kripke models, since it matches the expressive power of the modal language rather well. On the other hand, bisimulation has provided a characterization of the modal fragment of first-order logic (*FOL*). Modal formulas can be translated into formulas of *FOL*, and it turns out (see [5] and lemma 2) that the *modal fragment* of *FOL* is precisely its bisimulation-invariant fragment.

This line of investigation and the two main results mentioned can be extended from modal logic to Propositional Dynamic Logic (*PDL*) [12, 16], a logic where the modalities are indexed by programs. Programs can be constructed from atomic programs using a number of program operations such as sequential composition, iteration, etc., and like modal formulas, *PDL*-formulas are bisimulation-invariant. Secondly, iteration-free *PDL*-programs can be translated into *FOL* as well, raising the question how to characterize the *FOL*-fragment which (translations of) *PDL*-programs define. In [6], such a result has been obtained: The *program-fragment* of *FOL* can be characterized as its bisimulation-safe fragment,

P. Clote and H. Schwichtenberg (Eds.): CSL 2000, LNCS 1862, pp. 485–496, 2000.

where roughly speaking a program is safe for bisimulation if it preserves bisimulation. This result shows that if we take bisimulation as our notion of process equivalence and *FOL* as our language, the program operations provided by *PDL* are complete, i.e. no additional program operations will allow us to construct new programs. This result has been extended to monadic second-order logic in [13].

In this paper, we carry the investigation one step further, moving from nondeterministic programs (i.e. 1-player games) to 2-player games. In the program specification literature, such a move has been useful to obtain intermediate nonimplementable specifications which contain demonic as well as angelic choices [2, 3]. A formalism such as the refinement calculus models programs and specifications as predicate transformers, i.e. functions which map postconditions to weakest preconditions. This notion is general enough to model games as well as programs, and it is the semantic foundation of Game Logic (*GL*), introduced in [18]. In *GL*, the program operations of *PDL* are extended with a new construct called *dual*. In the terminology of games, this operation introduces a role switch between the players.

After introducing game models and *GL* in the next section, section 3 introduces bisimulation for game models. The first main result of this paper (proposition 1) shows that *GL*-formulas are invariant and *GL*-operations safe for bisimulation. Starting from section 4, we focus on a special class of models, Kripke models. For Kripke models, the generalized notion of bisimulation coincides with standard bisimulation and *GL* becomes a fragment of the modal μ-calculus which can express properties requiring multiple nested fixpoints. Section 5 is devoted to the second main result (proposition 2): Over Kripke models, iteration-free games (like programs) can be translated into *FOL*, thus defining the *game-fragment* of *FOL*. The result demonstrates that this fragment is precisely the monotonic bisimulation-safe fragment of *FOL*.

2 Syntax and Semantics of Game Logic

GL is a logic to reason about winning strategies in strictly competitive determined games between two players who we shall call Angel and Demon. For a game expression γ, the formula $\langle\gamma\rangle\varphi$ will express that Angel has a strategy in game γ for achieving φ, i.e. he can guarantee that the terminal position reached after γ has been played satisfies φ. Similarly, $[\gamma]\varphi$ will express that Demon has a strategy in game γ for achieving φ.

GL provides a number of operations which allow for the construction of complex games: A test game $\varphi?$ consists of checking through a neutral arbiter whether proposition φ holds at that state. If it does, nothing happens (i.e. another game can be played) and otherwise, Demon wins. The game $\gamma_1 \cup \gamma_2$ gives Angel the choice of playing γ_1 or γ_2. The sequential composition $\gamma_1;\gamma_2$ of two games consists of first playing γ_1 and then γ_2, and in the iterated game γ^*, Angel can choose how often to play γ, possibly not at all. More precisely, after each

play of γ, Angel can decide whether or not to play γ another time, but γ may not be played infinitely often (in that case, Demon wins).

In order to introduce interaction between the players, *GL* adds an operator *dual* for role interchange: Playing the dual game γ^{d} is the same as playing γ with the roles of the players reversed, i.e. any choice made by Angel in γ will be made by Demon in γ^{d} and vice versa.

Formally, the language of *GL* consists of two sorts, games and propositions. Given a set of atomic games Γ_0 and a set of atomic propositions Φ_0, games γ and propositions φ can have the following syntactic forms, yielding the set of games Γ and the set of propositions/formulas Φ:

$$\gamma := g \mid \varphi? \mid \gamma;\gamma \mid \gamma \cup \gamma \mid \gamma^* \mid \gamma^{\mathsf{d}}$$
$$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\gamma\rangle\varphi$$

where $p \in \Phi_0$ and $g \in \Gamma_0$. As usual, we define $\top := \neg\bot$, $[\gamma]\varphi := \neg\langle\gamma\rangle\neg\varphi$, $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \to \psi := \neg\varphi \vee \psi$ and $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$.

As for the semantics, given a signature (Φ_0, Γ_0) of atomic propositions and atomic games, a game model (also called neighborhood model or minimal model, see [9]) $\mathcal{I} = (S, \{N_g \mid g \in \Gamma_0\}, \{V_p \mid p \in \Phi_0\})$, consists of a set of states S, a valuation for each propositional letter $p \in \Phi_0$ such that $V_p \subseteq S$, and a function $N_g : \mathcal{P}(S) \to \mathcal{P}(S)$ for every atomic game $g \in \Gamma_0$. We require monotonicity, i.e. $X \subseteq Y$ implies $N_g(X) \subseteq N_g(Y)$ for all $g \in \Gamma_0$.

Intuitively, we can think of every state s as being associated with a 2-player game tree for every atomic game $g \in \Gamma_0$. Every terminal position of such a game tree is associated with a state $t \in S$. Since generally both players will have choices in the game, a player will usually not be able to force a particular state to come about at the end of the game. Rather, all he can do is force the outcome to lie in a particular set $Y \subseteq S$, and the game model specifies the sets of states which Angel can force, i.e. $s \in N_g(Y)$ holds if Angel has a strategy for ending up at a terminal position whose associated state is in Y. Given this interpretation, the monotonicity requirement is a natural one: If Angel has a strategy to bring about a state in Y, then that strategy trivially brings about a state in Y' for every $Y' \supseteq Y$.

The semantics of formulas and games is then defined by simultaneously extending V and N to non-atomic cases:

$$\begin{array}{ll} V_\bot = \emptyset & N_{\alpha;\beta}(X) = N_\alpha(N_\beta(X)) \\ V_{\neg\varphi} = \overline{V_\varphi} & N_{\alpha^{\mathsf{d}}}(X) = \widetilde{N_\alpha}(\overline{X}) \\ V_{\varphi\vee\psi} = V_\varphi \cup V_\psi & N_{\alpha\cup\beta}(X) = N_\alpha(X) \cup N_\beta(X) \\ V_{\langle\gamma\rangle\varphi} = N_\gamma(V_\varphi) & N_{\varphi?}(X) = V_\varphi \cap X \\ & N_{\alpha^*}(X) = \bigcap\{Y \subseteq S \mid X \cup N_\alpha(Y) \subseteq Y\} \end{array}$$

By induction, all N_α can be shown to be monotonic, and hence the operation $f_{\alpha,X}(Y) = X \cup N_\alpha(Y)$ will be monotonic as well. Thus, by the Knaster-Tarski theorem, $N_{\alpha^*}(X)$ is the least fixpoint of $f_{\alpha,X}(Y)$:

$$N_{\alpha^*}(X) = \mu Y.f_{\alpha,X}(Y) = \mu Y.X \cup N_\alpha(Y)$$

i.e. $f_{\alpha,X}(N_{\alpha^*}(X)) = N_{\alpha^*}(X)$ and for every $Z \subseteq S$, if $f_{\alpha,X}(Z) = Z$ then $N_{\alpha^*}(X) \subseteq Z$.

Finally, we say that φ is true in $\mathcal{I} = (S, \{N_g | g \in \Gamma_0\}, \{V_p | p \in \Phi_0\})$ at $s \in S$ (notation: $\mathcal{I}, s \models \varphi$) iff $s \in V_\varphi$. Some more standard terminology: φ is valid in $\mathcal{I}$ (denoted as $\mathcal{I} \models \varphi$) iff $V_\varphi = S$, and φ is valid (denoted as $\models \varphi$) iff it is valid in all models. φ and ψ are equivalent iff $\models \varphi \leftrightarrow \psi$. Lastly, ψ is a (global) consequence of φ (denoted as $\varphi \models \psi$) iff for all models $\mathcal{I}$, if $\mathcal{I} \models \varphi$ then $\mathcal{I} \models \psi$.

3 Bisimulation for Game Models

Bisimulation provides an answer to the question when two models or processes should be considered the same. Different criteria may come to mind depending on what aspects of the models one is interested in. If only interested in observable properties of processes, one may choose for finite-trace equivalence, but if interested in mathematical structure, one may choose isomorphism. These equivalence notions (see e.g. [7] for an overview) partition the class of models into equivalence classes, and one may order equivalence notions according to how fine-grained the induced partition is. While finite-trace equivalence is often considered as too coarse and isomorphism as too fine, bisimulation is situated between these two extremes.

As it stands, bisimulation cannot be applied to the game models of *GL* since these models are not processes. As will be discussed in the next section, the following definition generalizes the standard notion of bisimulation to the more general models used for *GL*. In a different context, this modification of bisimulation has been proposed to deal with concurrency in [4].

Definition 1 (Bisimulation). *Let* $\mathcal{I} = (S, \{N_g | g \in \Gamma_0\}, \{V_p | p \in \Phi_0\})$ *and* $\mathcal{I}' = (S', \{N'_g | g \in \Gamma_0\}, \{V'_p | p \in \Phi_0\})$ *be two models. Then* $\sim \subseteq S \times S'$ *is a bisimulation between* $\mathcal{I}$ *and* $\mathcal{I}'$ *iff for any* $s \sim s'$ *we have*

1. *For all* $p \in \Phi_0$*:* $s \in V_p$ *iff* $s' \in V'_p$
2. *For all* $g \in \Pi_0$*: If* $s \in N_g(X)$ *then* $\exists X' \subseteq S'$ *such that* $s' \in N'_g(X')$ *and* $\forall x' \in X' \, \exists x \in X : x \sim x'$.
3. *For all* $g \in \Pi_0$*: If* $s' \in N'_g(X')$ *then* $\exists X \subseteq S$ *such that* $s \in N_g(X)$ *and* $\forall x \in X \, \exists x' \in X' : x \sim x'$.

Two states $s \in S$ *and* $s' \in S'$ *are bisimilar iff there is a bisimulation* $\sim$ *such that* $s \sim s'$*. If we want to make the underlying models explicit, we will write* $(\mathcal{I}, s) \sim (\mathcal{I}', s')$.

The notions of invariance and safety generalize the bisimulation clauses from atomic to general formulas and games.

Definition 2 (*GL*-Invariance & Safety). *A GL-formula* φ *is invariant for bisimulation if for all models* $\mathcal{I}$ *and* $\mathcal{I}'$*,* $(\mathcal{I}, s) \sim (\mathcal{I}', s')$ *implies* $\mathcal{I}, s \models \varphi \Leftrightarrow \mathcal{I}', s' \models \varphi$*. A GL-game* γ *is safe for bisimulation if for all models* $\mathcal{I}$ *and* $\mathcal{I}'$*,* $(\mathcal{I}, s) \sim (\mathcal{I}', s')$ *implies (1) if* $s \in N_\gamma(X)$ *then* $\exists X' \subseteq S'$ *such that* $s' \in N'_\gamma(X')$

and $\forall x' \in X' \exists x \in X : x \sim x'$, *and (2) if* $s' \in N'_\gamma(X')$ *then* $\exists X \subseteq S$ *such that* $s \in N_\gamma(X)$ *and* $\forall x \in X \exists x' \in X' : x \sim x'$.

As an equivalence notion for game models, bisimulation requires that if Angel can guarantee φ in game g in one model, he must be able to guarantee something at least as strong in the other model. If this were not the case, the two models could be distinguished by playing g, since Angel can achieve more in one model than in the other. The following result shows that GL is sound for bisimulation equivalence, i.e. not too expressive: Bisimilar states cannot be distinguished by formulas of the language (invariance), and the game constructions provided do not produce games which can distinguish bisimilar states either (safety).

Proposition 1. *All GL-formulas are invariant for bisimulation, and all GL-games are safe for bisimulation.*

Proof. We prove invariance and safety by simultaneous induction on Φ and Γ. By definition, atomic games (formulas) are safe (invariant) for bisimulation.Consider two models $\mathcal{I} = (S, \{N_g | g \in \Gamma_0\}, \{V_p | p \in \Phi_0\})$ and $\mathcal{I}' = (S', \{N'_g | g \in \Gamma_0\}, \{V'_p | p \in \Phi_0\})$. For non-atomic formulas, the boolean cases are immediate and we shall only show one direction of invariance for $\langle\gamma\rangle\varphi$. If $\mathcal{I}, s \models \langle\gamma\rangle\varphi$, $s \in N_\gamma(V_\varphi)$ and so (by safety induction hypothesis for γ) there is some X' such that $s' \in N'_\gamma(X')$ and for all $x' \in X'$ there is some $x \in V_\varphi$ such that $x \sim x'$. By invariance induction hypothesis for φ, this means that $X' \subseteq V'_\varphi$, and so by monotonicity, $s' \in N'_\gamma(V'_\varphi)$, which establishes that $\mathcal{I}', s' \models \langle\gamma\rangle\varphi$.

As for proving that the game constructions of GL are safe for bisimulation, consider first the case of test $\varphi?$: If $s \in N_{\varphi?}(X) = V_\varphi \cap X$, let $X' := \{x' | \exists x \in X : x \sim x'\}$, where $\sim$ denotes the bisimulation as usual. Then $s' \in N'_{\varphi?}(X')$ by induction hypothesis (1.) for φ, and for all $x' \in X'$ there is some $x \in X$ such that $x \sim x'$, simply by definition of X'.

For union, if $s \in N_{\alpha\cup\beta}(X)$ we can assume w.l.o.g. that $s \in N_\alpha(X)$ and apply the induction hypothesis, i.e. for some X', we have $s' \in N'_\alpha(X')$ and hence also $s' \in N'_{\alpha\cup\beta}(X')$.

For composition, suppose that $s \in N_\alpha(N_\beta(X))$. Using the induction hypothesis for α, there is some Y' such that $s' \in N'_\alpha(Y')$ and for all $y' \in Y'$ there is a $u \in N_\beta(X)$ such that $u \sim y'$. Now let $X' := \{x' | \exists x \in X : x \sim x'\}$. We must show that $s' \in N'_\alpha(N'_\beta(X'))$. For this, it suffices by monotonicity to show that $Y' \subseteq N'_\beta(X')$. So suppose that $y' \in Y'$, i.e. for some $u \in N_\beta(X)$ we have $u \sim y'$. Using the induction hypothesis for β, there is some V' such that $y' \in N'_\beta(V')$ and for all $v' \in V'$ there is some $x \in X$ such that $x \sim v'$. Hence $V' \subseteq X'$ and so by monotonicity, $y' \in N'_\beta(X')$

Dual: Suppose $s \in N_{\alpha^d}(X)$, i.e. $s \notin N_\alpha(\overline{X})$. Again, let $X' := \{x' | \exists x \in X : x \sim x'\}$. It is sufficient to show that $s' \notin N'_\alpha(\overline{X'})$. Suppose by reductio the contrary. Then there is some Z with $s \in N_\alpha(Z)$ and for all $z \in Z$ there is some $x' \notin X'$ such that $z \sim x'$. From this it follows that $Z \subseteq \overline{X}$, so by monotonicity $s \in N_\alpha(\overline{X})$, a contradiction.

Iteration: Let $X' := \{x' | \exists x \in X : x \sim x'\}$ and $Z := \{z | \forall z' : z \sim z' \Rightarrow z' \in N'_{\alpha^*}(X')\}$. It is sufficient to show that $N_{\alpha^*}(X) \subseteq Z$, and given the definition of

$N_{\alpha^*}(X)$ as a least fixpoint, it suffices to show that $X \cup N_\alpha(Z) \subseteq Z$. Supposing that $x \in X$ and for some x' we have $x \sim x'$, we have $x' \in X' \subseteq N'_{\alpha^*}(X')$. On the other hand, suppose that $x \in N_\alpha(Z)$ and $x \sim x'$. Then by induction hypothesis, there is some Z' such that $x' \in N'_\alpha(Z')$ and for all $z' \in Z'$ there is some $z \in Z$ such that $z \sim z'$. But then $Z' \subseteq N'_{\alpha^*}(X')$, and so by monotonicity $x' \in N'_\alpha(N'_{\alpha^*}(X')) \subseteq N'_{\alpha^*}(X')$ which completes the proof. □

4 Games on Kripke Models I: μ-Calculus

In the remaining part of this paper, we shall look at a special class of game models, namely Kripke models. A Kripke model $\mathcal{I} = (S, \{R_g | g \in \Gamma_0\}, \{V_p | p \in \Phi_0\})$ differs from a game model in providing an accessibility relation $R_g \subseteq S \times S$ for every atomic game $g \in \Gamma_0$. In Kripke models, atomic games are particularly simple since they are 1-player games. For each atomic game, all choices within that game are made by Angel so that Angel has complete freedom in determining which terminal position will be reached. Thus, sR_gt will hold if when playing game g at state s, t is a possible final state. To obtain the corresponding game model from a Kripke model, let $N_g(X) = \{s \in S | \exists t \in X : sR_gt\}$. Under this correspondence, one can easily verify that for Kripke models, definition 1 indeed reduces to the following standard notion of bisimulation:

Definition 3 (Bisimulation for Kripke models). *Let $\mathcal{I} = (S, \{R_g | g \in \Gamma_0\}, \{V_p | p \in \Phi_0\})$ and $\mathcal{I}' = (S', \{R'_g | g \in \Gamma_0\}, \{V'_p | p \in \Phi_0\})$ be two Kripke models. Then $\sim \subseteq S \times S'$ is a bisimulation between $\mathcal{I}$ and $\mathcal{I}'$ iff for any $s \sim s'$ we have*

1. *For all $p \in \Phi_0$: $s \in V_p$ iff $s' \in V'_p$*
2. *For all $g \in \Gamma_0$: If sR_gt, then there is a $t' \in S'$ such that $s'R'_gt'$ and $t \sim t'$.*
3. *For all $g \in \Gamma_0$: If $s'R'_gt'$, then there is a $t \in S$ such that sR_gt and $t \sim t'$.*

Two well-known languages for describing Kripke models are *PDL* and the modal μ-calculus. The language of *PDL* differs from the language of *GL* only in not having the dual-operator available. Since this operator was responsible for introducing interaction between the players, all games which can be constructed within *PDL* will be 1-player games, i.e. nondeterministic programs.

The μ-calculus introduces fixpoint operators into the modal language, yielding a logic which is strictly more expressive than *PDL* (see [15]). Besides propositional constants Φ_0, the language contains propositional variables $X, Y, \ldots \in Var$ and the set of formulas is defined inductively as

$$\varphi := \bot \mid p \mid X \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\gamma_0\rangle\varphi \mid \mu X.\varphi$$

where $p \in \Phi_0$, $\gamma_0 \in \Gamma_0$, $X \in Var$ and in $\mu X.\varphi$, X occurs *strictly positively* in φ, i.e. every free occurrence of X in φ occurs under an even number of negations. Note that in contrast to *GL*, modalities are always atomic in the μ-calculus.

Formulas of the μ-calculus are interpreted over Kripke models as before (using the corresponding game model), but a variable assignment $v : Var \to \mathcal{P}(S)$ is needed to interpret variables. The semantics of the fixpoint formula is given by

$$V^v_{\mu X.\varphi} = \bigcap \{T \subseteq S | V^{v[X:=T]}_\varphi \subseteq T\}$$

where $V_\varphi^{v[X:=T]}$ differs from V_φ^v in assigning T to variable X. Since φ was assumed to be strictly positive in X, monotonicity is guaranteed and $\mu X.\varphi$ denotes the least fixpoint of the operation associated with $\varphi(X)$.

Inspecting the semantics of *GL*, one can easily translate *GL*-formulas into equivalent μ-calculus formulas, demonstrating that *GL* is a variable-free fragment of the μ-calculus. While a characterization of the precise expressiveness of this fragment is still lacking, some preliminary observations can be made: *GL* is strictly more expressive than *PDL*, since *GL* can express the existence of an infinite a-path by the formula

$$\mu X.[a]X \;=\; \langle (a^d)^* \rangle \bot$$

which cannot be expressed in *PDL* (see [15]). More complex properties such as "on some path p occurs infinitely often" ($EF^\infty p$ in CTL^* notation) can also be expressed (we assume that $\Gamma_0 = \{a\}$):

$$\nu X.\mu Y.\langle a \rangle((p \wedge X) \vee Y) = [((a^*; a; p?)^d)^*]\top$$

where $\nu X.\varphi$ abbreviates $\neg\mu X.\neg\varphi(\neg X)$ and yields the greatest fixpoint of $\varphi(X)$. More generally, if we let $g_0 = a$ and $g_{n+1} = (g_n^d)^*$, the μ-calculus translation of $\langle g_n \rangle \bot$ will be a formula of alternation depth n, so that *GL* formulas cover all levels of the alternation hierarchy as defined in [10].

5 Games on Kripke Models II: First-Order Logic

It is well-known that modal logic and *PDL* without iteration can be translated into *FOL*. In spite of the second-order appearance of Game Logic, a translation into *FOL* is possible here as well: The signature contains a unary relation symbol V_p for every propositional letter $p \in \Phi_0$, and a binary relation symbol R_g for every atomic game $g \in \Gamma_0$. Furthermore, we allow for second-order variables $\mathsf{X}, \mathsf{Y}, \ldots$ as well. Thus, the unary relation symbols now comprise constants as well as variables. As will become clear later, we will not quantify over these variables but only use them as a matter of convenience to serve as place-holders for substitution; hence, we can still consider the language to be first-order. We define the translation function $^\circ$ which maps a *GL*-formula φ to a *FOL*-formula with one free variable x, and an iteration-free *GL*-game γ to a *FOL*-formula with two free variables x and Y.

$$\begin{aligned}
p^\circ &= \mathsf{V}_p\mathsf{x} \quad \text{for } p \in \Phi_0 & g^\circ &= \exists \mathsf{z}(\mathsf{x}\mathsf{R}_g\mathsf{z} \wedge \mathsf{Y}\mathsf{z}) \quad \text{for } g \in \Gamma_0 \\
(\neg\varphi)^\circ &= \neg\varphi^\circ & (\varphi?)^\circ &= \varphi^\circ \wedge \mathsf{Y}\mathsf{x} \\
(\varphi \vee \psi)^\circ &= \varphi^\circ \vee \psi^\circ & (\alpha \cup \beta)^\circ &= \alpha^\circ \vee \beta^\circ \\
(\langle\gamma\rangle\varphi)^\circ &= \gamma^\circ[\mathsf{Y} := \varphi^\circ] & (\alpha;\beta)^\circ &= \alpha^\circ[\mathsf{Y} := \beta^\circ] \\
& & (\alpha^\mathsf{d})^\circ &= \neg\alpha^\circ[\mathsf{Y} := \neg\mathsf{Y}\mathsf{x}]
\end{aligned}$$

In this definition, substitution for second-order variables is used as follows: Given two *FOL*-formulas δ and ξ where ξ contains exactly one free first-order variable, say x, $\delta[\mathsf{Y} := \xi]$ denotes the result of replacing every occurrence $\mathsf{Y}t$ in

δ by $\xi[\mathsf{x} := t]$. As an example, $\exists \mathsf{z}(\mathsf{x}\mathsf{R}_g\mathsf{z} \wedge \mathsf{Y}\mathsf{z})[\mathsf{Y} := \neg\mathsf{Y}\mathsf{x}]$ yields $\exists \mathsf{z}(\mathsf{x}\mathsf{R}_g\mathsf{z} \wedge \neg\mathsf{Y}\mathsf{z})$. Some more remarks on notation: $\varphi(x_1, \ldots, x_n)$ refers to a formula φ whose free variables (first- and second-order) are among $x_1, \ldots, x_n$. When a formula has been introduced in this way, $\varphi(t_1, \ldots, t_n)$ denotes $\varphi[x_1 := t_1, \ldots, x_n := t_n]$, i.e. the simultaneous substitution of t_i for x_i in φ.

Regarding the semantics, we can interpret a Kripke model $\mathcal{I} = (S, \{R_g | g \in \Gamma_0\}, \{V_p | p \in \Phi_0\})$ as a first-order model in the obvious way, taking R_g as the interpretation of R_g, and interpreting V_p as V_p. For a unary predicate symbol V_p and $X \subseteq S$, let $\mathcal{I}_{p:=X}$ be the model which is the same as $\mathcal{I}$ except that $V_p = X$. Given a model $\mathcal{I}$, states $s_1, \ldots, s_m \in S$, sets of states $S_1, \ldots, S_n \subseteq S$ and a *FOL*-formula $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_m, \mathsf{X}_1, \ldots, \mathsf{X}_n)$, we write $\mathcal{I} \models \varphi[s_1, \ldots, s_m, S_1, \ldots, S_n]$ to denote that φ is true in $\mathcal{I}$ according to the standard *FOL* semantics when x_i is assigned the value s_i and X_i the value S_i.

The following result states the semantic correctness of the translation function.

Lemma 1. *For all GL-formulas φ, games γ and Kripke models $\mathcal{I} = (S, \{R_g | g \in \Gamma_0\}, \{V_p | p \in \Phi_0\})$: $\mathcal{I}, s \models \varphi$ iff $\mathcal{I} \models \varphi^\circ[s]$ and $s \in N_\gamma(X)$ iff $\mathcal{I} \models \gamma^\circ[s, X]$.*

As with the safety result for program constructions, the safety result for game constructions makes use of the characterization of the modal fragment of *FOL* as its bisimulation-invariant fragment. The definition of invariance and safety (definition 2) which was phrased for *GL* has its natural first-order analogue:

Definition 4 (*FOL*-Invariance & Safety). *A FOL-formula $\varphi(\mathsf{x})$ is invariant for bisimulation if for all models $\mathcal{I}$ and $\mathcal{I}'$, $(\mathcal{I}, s) \sim (\mathcal{I}', s')$ implies that $\mathcal{I} \models \varphi[s]$ iff $\mathcal{I}' \models \varphi[s']$. A first-order formula $\varphi(\mathsf{x}, \mathsf{Y})$ is safe for bisimulation if for all models $\mathcal{I}$ and $\mathcal{I}'$, $(\mathcal{I}, s) \sim (\mathcal{I}', s')$ implies (1) if $\mathcal{I} \models \varphi[s, T]$ then there is some T' such that $\mathcal{I}' \models \varphi[s', T']$ and for all $t' \in T'$ there is some $t \in T$ such that $t \sim t'$, and (2) if $\mathcal{I}' \models \varphi[s', T']$ then there is some T such that $\mathcal{I} \models \varphi[s, T]$ and for all $t \in T$ there is some $t' \in T'$ such that $t \sim t'$.*

By a *modal formula* we mean a *GL*-formula which only contains atomic games (i.e. also no tests). The classic result from [5] can now be stated as follows:

Lemma 2. *A FOL-formula $\varphi(\mathsf{x})$ is invariant for bisimulation iff it is equivalent to the translation of a modal formula.*

For the rest of this section, we will assume that games are iteration-free. Call a *FOL*-formula $\varphi(\mathsf{x}, \mathsf{Y})$ *monotonic* iff for all Kripke models $\mathcal{I}$ and states s, $\mathcal{I} \models \varphi[s, X]$ implies $\mathcal{I} \models \varphi[s, X']$ for every $X \subseteq X'$. Similarly, call a modal formula φ *monotonic in p* iff for all Kripke models $\mathcal{I}$ and states s, $\mathcal{I}_{p:=X}, s \models \varphi$ implies $\mathcal{I}_{p:=X'}, s \models \varphi$ for every $X \subseteq X'$. Lastly, let $Pos(\varphi)$ ($Neg(\varphi)$) be the set of atomic propositions which occur positively (negatively) in φ, i.e. under an even (odd) number of negations. Thus, formula φ is strictly positive (negative) in p iff $p \notin Neg(\varphi)$ ($p \notin Pos(\varphi)$).

The final lemma needed relates the syntactic notion of positivity to the semantic notion of monotonicity. It makes use of the Lyndon interpolation theorem for modal logic (see e.g. [17]) and the global deduction theorem (taken from [11]).

Lemma 3 (Lyndon Interpolation Theorem). *If $\models \alpha \to \beta$ for modal formulas α, β, then there exists a modal formula γ such that (1) $\models \alpha \to \gamma$, (2) $\models \gamma \to \beta$, (3) $Pos(\gamma) \subseteq Pos(\alpha) \cap Pos(\beta)$, and (4) $Neg(\gamma) \subseteq Neg(\alpha) \cap Neg(\beta)$.*

Lemma 4 (Global Deduction Theorem). *For modal formulas δ and γ, $\delta \models \gamma$ iff there is some $n > 0$ such that $\models (\Box^1\delta \wedge \ldots \wedge \Box^n\delta) \to \gamma$, where each $\Box^i$ represents a possibly empty sequence of universal modalities labeled by (possibly different) atomic games.*

Lemma 5. *A modal formula φ is monotonic in p iff it is equivalent to a modal formula strictly positive in p.*

Proof. One can easily check by induction that strictly positive modal formulas are monotonic, so we shall only prove the other direction. If $\varphi(p)$ is monotonic in p, then taking a proposition letter q not occurring in φ, we have $p \to q \models \varphi(p) \to \varphi(q)$ (recall that semantic consequence was defined globally). By lemma 4, we know that

$$(\Box^1(p \to q) \wedge \ldots \wedge \Box^n(p \to q)) \to (\varphi(p) \to \varphi(q))$$

is valid, and as a consequence,

$$\varphi(p) \to ((\Box^1(p \to q) \wedge \ldots \wedge \Box^n(p \to q)) \to \varphi(q))$$

is also valid. By lemma 3, this implies that

$$\varphi(p) \to \gamma \quad \text{and} \quad \gamma \to ((\Box^1(p \to q) \wedge \ldots \wedge \Box^n(p \to q)) \to \varphi(q))$$

are valid, for some modal formula γ which does not contain q and which is strictly positive in p. The second conjunct implies that $\gamma \to \varphi(p)$ is valid: For suppose $\mathcal{I}, s \models \gamma$ and $X = \{t | \mathcal{I}, t \models p\}$. Then since γ does not contain q, $\mathcal{I}_{q:=X}, s \models \gamma$. From this it follows that $\mathcal{I}_{q:=X}, s \models \varphi(q)$ and hence $\mathcal{I}, s \models \varphi(p)$. Thus, φ is equivalent to γ, a modal formula strictly positive in p. □

The main lemma we need for our safety result relates monotonic modal formulas to *GL*-formulas of a special kind.

Lemma 6. *Every modal formula φ which is monotonic in p is equivalent to a GL-formula $\langle\gamma\rangle p$, where γ is a game which does not contain p.*

Proof. We prove by induction that every modal formula φ which is strictly positive (negative) in p is equivalent to a *GL*-formula $\langle\gamma\rangle p$ ($\neg\langle\gamma\rangle p$), where γ does not contain p. Then the result follows by lemma 5. The following table provides the equivalent *GL*-formulas for every modal formula φ depending on whether φ is strictly positive or strictly negative in p.

modal formula	str. pos/neg	*GL*-formula	ind. hyp.
p	*pos*	$\langle \top ? \rangle p$	–
$q \neq p$	*pos*	$\langle q?; \bot ?^{\mathsf{d}} \rangle p$	–
$q \neq p$	*neg*	$\neg \langle q?^{\mathsf{d}}; \bot ? \rangle p$	–
$\neg \varphi$	*pos*	$\langle \gamma \rangle p$	$\models \varphi \leftrightarrow \neg \langle \gamma \rangle p$
$\neg \varphi$	*neg*	$\neg \langle \gamma \rangle p$	$\models \varphi \leftrightarrow \langle \gamma \rangle p$
$\varphi_1 \vee \varphi_2$	*pos*	$\langle \gamma_1 \cup \gamma_2 \rangle p$	$\models \varphi_i \leftrightarrow \langle \gamma_i \rangle p$
$\varphi_1 \vee \varphi_2$	*neg*	$\neg \langle (\gamma_1{}^{\mathsf{d}} \cup \gamma_2{}^{\mathsf{d}})^{\mathsf{d}} \rangle p$	$\models \varphi_i \leftrightarrow \neg \langle \gamma_i \rangle p$
$\langle g \rangle \varphi$	*pos*	$\langle g; \gamma \rangle p$	$\models \varphi \leftrightarrow \langle \gamma \rangle p$
$\langle g \rangle \varphi$	*neg*	$\neg \langle g^{\mathsf{d}}; \gamma \rangle p$	$\models \varphi \leftrightarrow \neg \langle \gamma \rangle p$

□

Proposition 2. *A FOL-formula* $\varphi(\mathsf{x}, \mathsf{Y})$ *is equivalent to the translation of a GL-game iff it is safe for bisimulation and monotonic in* Y.

Proof. If $\varphi(\mathsf{x}, \mathsf{Y})$ is equivalent to the translation of a *GL*-game γ, then using lemma 1, φ will be monotonic in Y (because N_γ is monotonic) and safe for bisimulation (by proposition 1).

For the converse, assume that $\varphi(\mathsf{x}, \mathsf{Y})$ is monotonic and safe for bisimulation. Taking a new predicate symbol V_p which does not occur in φ, $\varphi(\mathsf{x}, \mathsf{V}_p)$ will be invariant for bisimulation. By lemma 2, $\varphi(\mathsf{x}, \mathsf{V}_p)$ is equivalent to the translation of a modal formula δ, i.e. $\models \varphi(\mathsf{x}, \mathsf{V}_p) \leftrightarrow \delta^\circ$. Since $\varphi(\mathsf{x}, \mathsf{Y})$ was monotonic, δ will be monotonic in p and by lemma 6, $\models \delta \leftrightarrow \langle \gamma \rangle p$ where γ is a *GL*-game which does not contain p, and so $\models \varphi(\mathsf{x}, \mathsf{V}_p) \leftrightarrow (\langle \gamma \rangle p)^\circ$. It can now be checked that $\models \varphi(\mathsf{x}, \mathsf{Y}) \leftrightarrow \gamma^\circ$: If $\mathcal{I} \models \varphi[s, X]$ then given that V_p does not occur in φ, $\mathcal{I}_{p:=X} \models \varphi(\mathsf{x}, \mathsf{V}_p)[s]$ and so $\mathcal{I}_{p:=X} \models (\langle \gamma \rangle p)^\circ[s]$. Since p does not occur in γ, this implies that $\mathcal{I} \models \gamma^\circ[s, X]$. The converse is proved along the same lines. □

On the one hand, proposition 2 provides a characterization result for the iteration-free games which can be constructed in Game Logic: *GL*-games are the monotonic bisimulation-safe formulas $\varphi(\mathsf{x}, \mathsf{V}_p)$ of first-order logic (we can simply replace the variable Y by a designated unary predicate constant V_p). In other words, the game-fragment of *FOL* is precisely the monotonic bisimulation-safe fragment. On the other hand, looking at the set of operations on games which *GL* provides, one may ask whether one could not add other natural operations to create new games (e.g. playing games in parallel), thus increasing the expressive power of the language. Proposition 2 demonstrates that if the new game operation is (1) first-order definable, (2) monotonic and (3) safe for bisimulation, then it is expressible in *GL* already. As argued before, requirements (2) and (3) are natural desiderata for games, i.e. they are minimal requirements for any alleged game operation, and so the operations of test, union, composition and dual are sufficient to construct all first-order definable games.

The result concerning bisimulation-safe programs from [6] can be reformulated to fit the present framework. Semantically, the difference between games

and programs lies in the difference between monotonicity and continuity: Call a *FOL*-formula $\varphi(\mathsf{x}, \mathsf{Y})$ *continuous* iff for all Kripke models $\mathcal{I}$ and states s, $\mathcal{I} \models \varphi[s, \bigcup_{X \in V} X]$ iff there is some $X \in V$ for which $\mathcal{I} \models \varphi[s, X]$ holds. Then the program-analogue of proposition 2 states that a *FOL*-formula $\varphi(\mathsf{x}, \mathsf{Y})$ is equivalent to the translation of a *GL*-program iff it is safe for bisimulation and continuous in Y, where *GL*-programs are dual-free *GL*-games. Thus, the dual operator makes all the difference between programs and games; without dual, we obtain all first-order definable programs, with dual, all first-order definable games.

6 Beyond First-Order Logic

The last two sections were concerned with Kripke models rather than game models in general. The reason for this restriction is that game models are rather unorthodox structures. We do not know of any logical languages besides non-normal modal logics and Game Logic which have been proposed for these structures. Consequently, this prevents an easy extension of the definability result of proposition 2 to *GL* over general game models.

Even for Kripke models, the translation into *FOL* carried out in the previous section relied on the restriction to iteration-free games. For programs, a stronger definability result covering iteration has been obtained in [13] which characterizes the class of monadic-second-order definable programs which are safe for bisimulation. The proof makes use of the fact that the bisimulation-invariant fragment of monadic second-order logic is the μ-calculus [14]. An extension of proposition 2 along these lines however would require a better understandin' of how exactly *GL* relates to the μ-calculus. As for the μ-calculus itself, mar fundamental properties were established only recently, such as completeness [19] the non-collapse of the alternation-hierarchy [8] and uniform interpolation [1], and others such as Lyndon interpolation are still open.

To summarize, the restriction of the scope of proposition 2 to *FOL* is due to the fact that *FOL* is one of the logics we know most about and is able to express the most fundamental game-operations. When moving to stronger languages one has different options available, always depending on the game constructions one is interested in. For besides playing a game iteratively, playing two games in parallel or interleaved might present another attractive game construction worth investigating in relation to bisimulation.

References

1. Giovanna D'Agostino. *Modal Logic and non-well-founded Set Theory: translation, bisimulation, interpolation.* PhD thesis, University of Amsterdam, 1998.
2. Ralph-Johan Back and Joakim von Wright. *Refinement Calculus - A systematic introduction.* Springer, 1998.
3. R.J.R. Back and J. von Wright. Games and winning strategies. *Information Processing Letters*, 53:165–172, 1995.
4. J. van Benthem, J. van Eijck, and V. Stebletsova. Modal logic, transition systems and processes. Computer Science Report CS-R9321, CWI, 1993.

5. Johan van Benthem. *Modal Correspondence Theory.* PhD thesis, University of Amsterdam, 1976.
6. Johan van Benthem. Program constructions that are safe for bisimulation. *Studia Logica*, 60(2):311–330, 1998.
7. Johan van Benthem and Jan Bergstra. Logic of transition systems. ILLC Prepublication Series CT-93-03, University of Amsterdam, 1993.
8. J. C. Bradfield. The modal mu-calculus alternation hierarchy is strict. In U. Montanari and V. Sassone, editors, *Proceedings of CONCUR '96*, volume 1119 of *LNCS*, pages 233–246. Springer, 1996.
9. Brian Chellas. *Modal Logic - An Introduction.* Cambridge University Press, 1980.
10. E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proceedings of the 1st IEEE Symposium on Logic in Computer Science*, pages 267–278, 1986.
11. Melvin Fitting. Basic modal logic. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of logic in artificial intelligence and logic programming*, volume 1, pages 365–448. Oxford University Press, 1993.
12. David Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II. D. Reidel Publishing Company, 1984.
13. Marco Hollenberg. *Logic and Bisimulation.* PhD thesis, University of Utrecht, 1998.
14. David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *Proceedings of CONCUR '96*, volume 1119 of *LNCS*, pages 263–277. Springer, 1996.
15. Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
16. Dexter Kozen and Jerzy Tiuryn. Logics of programs. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B. MIT Press, 1990.
17. Larisa Maksimova. Amalgamation and interpolation in normal modal logics. *Studia Logica*, 50(3-4):457–471, 1991.
18. Rohit Parikh. The logic of games and its applications. In Marek Karpinski and Jan van Leeuwen, editors, *Topics in the Theory of Computation*, volume 24 of *Annals of Discrete Mathematics*. Elsevier, 1985.
19. Igor Walukiewicz. A note on the completeness of Kozen's axiomatisation of the propositional μ-calculus. *Bulletin of Symbolic Logic*, 2(3):349–366, 1996.